Armaaruss Military Drone and Soldier Detection System
(Armaaruss Model Sys Version 1)

by Anthony of Boston

This document contains the javascript source code for the visual component of Armaaruss, which is a military detection program that can detect drones and enemy soldiers. The HTML code can simply be copy and pasted from the document. (download the document before copying and pasting) This is the first model sys version of Armaaruss, and contains new programming techniques for object detection. This code puts both aspects of drone and human detection into one application. Here is look at the applications's interface:



This drone and soldier detection javascript application has multiple functionalities. It comes with both a primary and secondary detection. The primary detection(the white bounding box) is the basic object detection using the webcam of one's device. However, the primary detection does not react to color calibration, meaning that the output is the same regardless of whether the screen or frame is bright or dark. The secondary detection's output is tied to color calibration, meaning that using the different filters used to alter the appearance the frame/video will affect the output of the secondary detection.

You can test this on the app by clicking the “Black” button. The screen will turn black, but primary detection can still be activated, while secondary detection can no longer display a bounding box. Secondary detection is functional, but cannot see or detect anything because the screen is black—it is essentially detecting a black image, while primary detection is still processing video output. The breakthrough here is that the javascript code for the secondary detection is treating the webcam video output as an image, and not a video. A good way to understand secondary detection is to imagine continuously uploading a new image file every second. With tensorflow’s default object detection code for detecting objects on various images, uploading a new image file every second is what one would have to achieve in order to match the processes of secondary detection. This application entitled “Armaaruss Military Drone and Soldier Detection System” manipulates tensorflow script to have every frame detected and treated as an image, with the bounding boxes rendered and removed with each frame. This is what makes Armaaruss’s vision component different from other object detection applications.

Thus, it is inferred that secondary detection can be used to detect object in different scenarios without having to re-train the models over and over again. This provides the possibility that drones can also be detected at night by applying the “invert” and other filters to color calibrate the frame that contains the video output. The primary detection box is “white”, while the secondary detection box is “red.”

This Armaaruss application, while not quite optimized for mobile use, comes with an aerial object detection system and intruder detection system that uses a beeping sound and voice to indicate when these have been detected. The aerial object detection system will beep when an aerial object is located. The longer an aerial object is hovering near you, the longer the beeping noise. For soldiers, this could mean that a drone is targeting them. Ideally, soldiers would use the app on their cell phones and attach the device to the top area of their vehicles or to their body while sleeping in the trenches. Keep mind that sim cards must be removed and cell phone wireless connectivity must remain "off" in combat environments. Before deployment, soldiers should connect to wifi and start the app. Once the app is started, a soldier can then disable wifi and leave the app running as he/she is deployed into a combat zone. For detecting aerial objects in combat, android phone should be mounted to the top of the backpack or top of the helmet.

In civilian environments, the cell phone, with wireless turned on, could be placed on rooftops. With internet access, a user could view the aerial scene remotely with facebook live
.



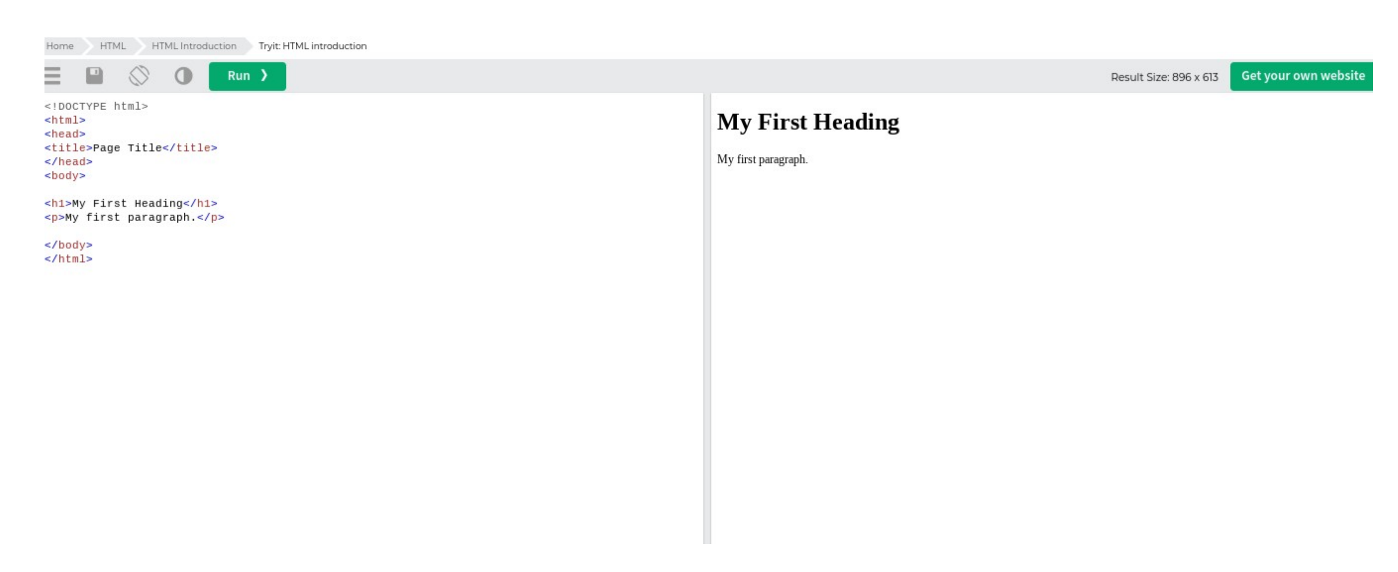


The Intruder Detection application will beep when an intruder is detected. The app will also emit voice alert, saying "intruder detected" upon detection of an intruder. This app allows phones to be mounted in various places, to which it can detect when an intruder is in the area. This can be used in clearing and counter insurgency operations. This is also useful for the civilian population against thugs and other criminal elements in urban environments. The app can prevent ambush attacks. The phone could be placed in the cracks of walls and other discreet locations. With internet access, a user could view the scene remotely with facebook live and see when intruders have gained unauthorized access.

"Intruder Detected"
"Intruder Detected"


"Intruder Detected"

For testing, one can copy and paste this html code at a website called https://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro



Just copy and paste the following HTML code on the left side in order to test the object, intruder, and drone detection on your webcam. (ends on page 52)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Armaaruss object detection using pre trained model in TensorFlow.js</title>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">



    <!-- Import the webpage's stylesheet -->
    <link rel="stylesheet" href="/style.css">



  </head>

  <style>

html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquote,pre,a,abbr,acronym,address,big,cite,code,del,dfn,em,img,ins,kbd,q,
s,samp,small,strike,strong,sub,sup,tt,var,b,u,i,center,dl,dt,dd,ol,ul,li,fieldset,form,label,legend,table,caption,tbody,tfoot,thead,tr,th,td,article,
aside,canvas,details,embed,figure,figcaption,footer,header,hgroup,menu,nav,output,ruby,section,summary,time,mark,audio,video{font-
size:100%;font:inherit;padding:0;border:0;margin:0;vertical-align:baseline z-index:12;}body{line-height:1}ol,ul{list-
style:none}blockquote,q{quotes:none}blockquote:before,blockquote:after,q:before,q:after{content:'';content:none}table{border-
collapse:collapse;border-
spacing:0}article,aside,details,figcaption,figure,footer,header,hgroup,menu,nav,section{display:block}.clear{clear:both}
.sticky{}.bypostauthor{}.wp-caption{}.wp-caption-text{}.gallery-caption{}.alignright{}.alignleft{}.aligncenter{}


textarea:focus, input:focus{outline: none; }
*:focus {outline: none;}


body {


  background-color: #999999;
}


.wrapper {


width: 100vw;
height: 100vh;
float: left;
box-sizing: border-box;
position: relative;


   }




#endec1{
left: 300px;
top: 400px;
}


#endec2{
left: 300px;
top: 400px;
}


#intru1{
font-size: 20px;
position: fixed;
left: 340px;
top: 500px;
-webkit-animation: fit 1s infinite;
  animation: fit 1s infinite;


}


#intru2{
font-size: 20px;
position: fixed;
left: 340px;
top: 500px;
-webkit-animation: fit 1s infinite;
  animation: fit 1s infinite;


}
```

```
#intru{
font-size: 20px;
position: fixed;
left: 340px;
top: 500px;
-webkit-animation: fit 1s infinite;
  animation: fit 1s infinite;


}





.title {
  width: 100%;
  height: 20vh;
  display: table;
  text-align: center;
  box-sizing: border-box;
}
.title h1 {
font-size: 50px;
color: #FFFFFF;
  display: table-cell;
  vertical-align: middle;
}


.vision {
width: 100%;
height: 80vh;
position: relative;
overflow: hidden;
    z-index: 10;
}


.stage {
width: 100%;
  height: 100%;
  position: absolute;
  top: 0;
  left: 0;
  right: 0;


background-size: cover;
background-repeat: no-repeat;
background-position: center;
}


.overlay {
width: 100%;
  height: 100%;
  position: relative;


background-repeat: repeat;
background-position: center;
}


.overlay .positionals {
width: 25%;
margin: 0 auto;
position: absolute;
bottom: 50px;
left: 50px;
text-align: left;
}
.overlay .positionals p {font-size: 12px;}


.overlay .model {
width: 25%;
margin: 0 auto;
position: absolute;
bottom: 50px;
right: 50px;
text-align: right;
}
.overlay .model p {font-size: 12px;}
```

```
.overlay .left {
width: 40%;
position: absolute;
top: 50px;
left: 50px;
}
.overlay .right {
width: 40%;
position: absolute;
top: 50px;
right: 50px;
text-align: right;
}
.overlay p {
font-size: 10px;
color: #FFFFFF;
margin: 0 auto;
}
.overlay .center {
width: 50%;
margin: 0 auto;
position: absolute;
bottom: 50px;
left: 0;
right: 0;
text-align: center;
}
.overlay .center p {font-size: 20px;}
.overlay .center p span {opacity: 1;}
span.letter1 {
-webkit-animation: letterone 1s infinite;
  animation: letterone 1s infinite;
}
span.letter2 {
-webkit-animation: lettertwo 1s infinite;
  animation: lettertwo 1s infinite;
}
span.letter3 {
-webkit-animation: letterthree 1s infinite;
  animation: letterthree 1s infinite;
}
span.letter4 {
-webkit-animation: letterfour 1s infinite;
  animation: letterfour 1s infinite;
}
span.letter5 {
-webkit-animation: letterfive 1s infinite;
  animation: letterfive 1s infinite;
}
span.letter6 {
  -webkit-animation: lettersix 0.75s infinite;
  animation: lettersix 0.75s infinite;
}



@-webkit-keyframes letterone {80% {opacity: 0;}}
@keyframes letterone {80% {opacity: 0;}}

@-webkit-keyframes lettertwo {85% {opacity: 0;}}
@keyframes lettertwo {85% {opacity: 0;}}

@-webkit-keyframes letterthree {90% {opacity: 0;}}
@keyframes letterthree {90% {opacity: 0;}}

@-webkit-keyframes letterfour {95% {opacity: 0;}}
@keyframes letterfour {95% {opacity: 0;}}

@-webkit-keyframes letterfive {100% {opacity: 0;}}
@keyframes letterfive {100% {opacity: 0;}}

@-webkit-keyframes lettersix {100% {opacity: 0;}}
@keyframes lettersix {100% {opacity: 0;}}


p.dimension1,
```

```
p.dimension2,
p.dimension3,
p.dimension4,
p.dimension5 {opacity: 0;}


p.dimension1.show,
p.dimension2.show,
p.dimension3.show,
p.dimension4.show,
p.dimension5.show {opacity: 1;}


p.dimension5.show {
-webkit-animation: fit 1s infinite;
  animation: fit 1s infinite;
}


p.dimension55.show {
-webkit-animation: fit 1s infinite;
  animation: fit 1s infinite;
}
@-webkit-keyframes fit {100% {opacity: 0;}}
@keyframes fit {100% {opacity: 0;}}






/*
----------
BELOW 1400
----------
*/
@media screen and (max-width: 1399px) {


  .overlay p {font-size: 30px;}


}


/*
----------
BELOW 1000
----------
*/
@media screen and (max-width: 999px) {


  .title h1 {font-size: 12px; font-weight: bold;}
  .overlay p {font-size: 12px; font-weight: bold;}


}
  body {


}


h1 {
  visibility:hidden;
}


 #title1 {
font-size: 12px;
font-weight: bold;
color: #ffffff;
top: 49px;
left: 10px;
position: fixed;


}



.videoView, .classifyOnClick {
  position: absolute;


   z-index: 100;
```

```
  cursor: pointer;

}

#liveView {

border: none;
z-index: 0;
position: fixed;
font-style: bold;
  color: #ff9853;

    min-width: 100%; min-height: 100%;
    width: auto; height: auto; z-index: 0;

    background-size: cover;

}

video {

}

 #webcamButton{
z-index: 10;
position: relative;

}

 .classifyOnClick1 p {
 border: 5px solid #ffffff;
z-index: 3;
position: fixed;
font-style: bold;
font-size: 20px;
  color: #ffffff;
  max-width: 80%; max-height: 100%;
 min-width: 70%; min-height: 100%;
  width: auto; height: auto;
     z-index: 2;
      margin-top: 5%;
   margin-left: 14%;
   margin-right: 10%;
   margin-bottom: 10%;
    background-size: cover;

}

.classifyOnClick p {
 border: 7px solid #ff0000;
z-index: 0;
position: fixed;
font-style: bold;
font-size: 20px;
  color: #ff0000;

   max-width: 80%; max-height: 100%;
 min-width: 70%; min-height: 100%;
  width: auto; height: auto;   z-index: 0;
   margin-top: 5%;
   margin-left: 14%;
   margin-right: 10%;
```

```
   margin-bottom: 10%;
    background-size: cover;


}

.classifyOnClick2 {

z-index: 11;
position: fixed;




}

#lefty{

top: 180px;

}

#righty{

top: 180px;

}



.highlighter1 {
  background: rgba(0, 255, 0, 0.25);
  border: 1px dashed #fff;
  z-index: 1;
  position: absolute;
}

.highlighter {

  background: rgba(0, 0, 0, 0);
  border: none;
  z-index: 4;
  position: absolute;
}

.classifyOnClick {
  z-index: 4;
}



canvas{


    zoom: 100%;

}

#endec {
right: 40px;

}

#demo{
top:73px;
```

```
left: 15px;
font-weight: bold;
font-size: 13px;
color: #ffffff;
  position: fixed;


z-index: 4;


}


#digital-clock{
top: 100px;
left: 10px;
font-weight: bold;
font-size: 13px;
color: #ffffff;
  position: fixed;


z-index: 4;


}


.classifyOnClick1 progress {
  width: 10%;
  height: 20px;
right: 50px;
top: 70px;
  position: fixed;
  z-index: 10;
}
.classifyOnClick1 progress.charging {
 border: 3px solid black;
right: 50px;
 position: fixed;
  z-index: 10;
}
.classifyOnClick1 progress.draining {
  border: 3px solid red;
 right: 50px;
  position: fixed;
   z-index: 10;
}




#batteryname {
right: 57px;
top: 55px;
  position: fixed;
  z-index: 10;
  font-weight: bold;
font-size: 13px;
color: #ffffff;
  position: fixed;


z-index: 4;



}


  </style>


  <body >
  <p id="batteryname"></p>


  <div class="classifyOnClick1">
 <progress  id="battery" value="0" max="100"></progress></div>
  <script>
  document.getElementById("batteryname").innerHTML = "Battery status";
  var progress = document.getElementById('battery');


navigator.getBattery().then(function(battery) {
  function updateChargeInfo(){
```

```
    progress.className = (battery.charging ? "charging" : "draining")
  }

  function updateLevelInfo(){
    progress.value = battery.level * 100;
  }

  battery.addEventListener('chargingchange', function(){
    updateChargeInfo();
  });

  battery.addEventListener('levelchange', function(){
    updateLevelInfo();
  });

  updateChargeInfo();
  updateLevelInfo();
});
</script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<!-- Font -->
<link href="https://fonts.googleapis.com/css?family=Inconsolata:700" rel="stylesheet">

<!-- Wrapper -->
<div class="wrapper">

    <!-- Title -->
<div class="title">
      <h1><span class="letter6">_</span></h1>
    </div>

<!-- Vision -->
<div class="vision">

<!-- Stage -->
<div class="stage"></div>

<!-- Overlay -->
<div class="overlay">

<!-- Positionals -->
<div id="leftbottom" class="positionals" style="position:fixed" >
<p>X <span class="positionx"></span></p>
<p>Y <span class="positiony"></span></p>
</div>

<!-- Model -->
<div id="rightbottom" class="model" style="position:fixed">
<p>Armaaruss Model</p>
<p>Sys Ver 1</p>
        <p>Mars 360 system</p>
</div>
<p  id="intru1"></p>
<p id="intru2"></p>
<p id="intru"></p>
<!-- Left -->
<div id="lefty" class="left" style="position:fixed" style="position:fixed"   >

<p >SYSTEM BOOT</p>
<p>ANALYSIS****</p>
<p id="randomnumber1"></p>
<p id="randomnumber2"></p>
<p id="randomnumber3"></p>

<br><br><br><br>

<p>ASSESSMENT</p>
<p>*****************</p>
<p>TOLERANCE LVL4</p>
<p>DETECT<span class="letter6">_</span></p>
```

```
</div>

<!-- Center -->
<div class="center">
<p>

</p>
</div>

<!-- Right -->
<div id="righty" class="right" style="position:fixed" >

<p>SCAN MODE <span id="randomscan">438894</span></p>
<p>RESET TO ACQUISITION</p>
<p>SPEECH GRADE: ****<span class="letter6">_</span></p>

<br><br><br><br>

<p>VISUAL SCAN</p>
<p>************</p>

<p id="endec" style="position:fixed" class="dimension5">SECONDARY DETECTION ACTIVE</p>



</div>

</div>

</div>

</div>
  <p id="demo"></p>

  <div class="classifyOnClick2" ><button  id="webcamButton3" onClick="enableCam()">Enable Webcam(all object detection)</button>
    <button id="webcamButton" onClick="invert()" >Invert</button>
      <button id="webcamButton1" onClick="intruderd()" >Turn on Intruder Detection</button>
       <button id="webcamButton2" onClick="aerialobjectd()" >Turn on Aerial Object Detection</button>
         <button id="webcamButton" onClick="grayscale()" >Turn on Grayscale</button>
          <button id="webcamButton" onClick="Reset()" >Reset Color</button>
           <button id="webcamButton" onClick=" invertandgrayscale()" >Invert and Grayscale</button>
           <button id="webcamButton" onClick=" invertandgrayscaleandcontrast()" >Invert, Grayscale and Contrast</button>

           <button id="webcamButton4" onClick=" prima()" >Turn Off Primary Detection</button>
            <button id="webcamButton4" onClick=" brightnessOff()" >Black</button>
          <button id="message"  >Stop Secondary Detection</button></div>
           </div>

  <p class="classifyOnClick1" id="demo">



   <div id="digital-clock"></div>



      <div class="classifyOnClick">
        <canvas id="myCanvas" style="filter:opacity(0%)" width=900 height=700  />
      </div>

<script>

  </script><script>
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
```

```
var img = new Image();
img.onload = function(){

      };
  img.crossOrigin = "Anonymous";
img.src = "https://yourimageshare.com/ib/ZfsEzBXqF0.png";
  </script>

      <p id="title1" class="classifyOnClick">Armaaruss Detection Systems</p>

     <div class="classifyOnClick1" id="liveView" >

        <video id="webcam" autoplay width=100% height=700 ></video>
      </div>

     <script> var beep = (function () {

var ctxClass = window.audioContext ||window.AudioContext || window.AudioContext || window.webkitAudioContext

var ctxs = new ctxClass();

return function (duration, type, finishedCallback) {

duration = +duration;

type = (type % 5) || 0;

if (typeof finishedCallback != "function") {

finishedCallback = function () {};

}

var osc = ctxs.createOscillator();
```

```
osc.type = type;

osc.connect(ctxs.destination);
if (osc.noteOn) osc.noteOn(0);
if (osc.start) osc.start();

setTimeout(function () {
if (osc.noteOff) osc.noteOff(0);
if (osc.stop) osc.stop();
finishedCallback();
}, duration);

};
})();

function textToSpeech() {
const speech = new SpeechSynthesisUtterance();
let voices = speechSynthesis.getVoices();

let convert = document.getElementById("intru1").innerHTML;

speech.text = convert;

speech.volume = 1;
speech.rate = 0.9;
speech.pitch = 0;

speech.voice = voices[3];

speechSynthesis.speak(speech);
}




function pause() {
```

```
window.speechSynthesis.pause();

}



function stop2() {

window.speechSynthesis.cancel();

}

function textToSpeech1() {

const speech1 = new SpeechSynthesisUtterance();

let voices1 = speechSynthesis.getVoices();


let convert1 = document.getElementById("intru2").innerHTML;


speech1.text = convert1;


speech1.volume = 1;

speech1.rate = 0.9;

speech1.pitch = 0;


speech1.voice = voices1[3];


speechSynthesis.speak(speech1);

}




</script>




    <!-- Import TensorFlow.js library -->
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js" type="text/javascript"></script>

    <!-- Load the coco-ssd model to use to recognize things in images -->
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/coco-ssd"></script>

    <!-- Import the page's JavaScript to do some stuff -->
    <script src="/script.js" defer></script>

    <script>
```

```
    const demosSection = document.getElementById('demos');

var model = undefined;

// Before we can use COCO-SSD class we must wait for it to finish
// loading. Machine Learning models can be large and take a moment to
// get everything needed to run.
cocoSsd.load().then(function (loadedModel) {
  model = loadedModel;
  // Show demo section now model is ready to use.
  demosSection.classList.remove('invisible');
});



/********************************************************************
// Demo 1: Grab a bunch of images from the page and classify them
// upon click.
********************************************************************/

// In this demo, we have put all our clickable images in divs with the
// CSS class 'classifyOnClick'. Lets get all the elements that have
// this class.




/********************************************************************
// Demo 2: Continuously grab image from webcam stream and classify it.
// Note: You must access the demo on https for this to work:
// https://tensorflow-js-image-classification.glitch.me/
********************************************************************/



const video = document.getElementById('webcam');
const liveView = document.getElementById('liveView');

var el = true;

/* var flipBack = document.querySelector(".flip-back"); */




function getVideo(el){

  navigator.mediaDevices.getUserMedia({
    video: {
/* facingMode: {
      exact: el?'user':'environment'
     } */
          facingMode: el?'user':'environment'
},
    audio: false
  }).then(d=>{
  (el===false)?video.classList.add("back"):video.classList.remove("back");
    video.srcObject = d;


  })
  .catch(err=>{
  var msg = 'Either your video cam is missing OR not working properly. Please check.';
  (err.name==='NotFoundError')?alert('Error name: '+err.name+'\nError msg: '+msg):alert('Error name: '+err.name+'\nError msg:
'+err.message);
  });
}

getVideo(el);
setInterval(function(){

  ctx.drawImage(video, 0, 0, video.clientWidth, video.clientHeight);
},0);

var stop = () => video.srcObject && video.srcObject.getTracks().map(t => t.stop());
```

```
// Check if webcam access is supported.
function hasGetUserMedia() {
  return !!(navigator.mediaDevices &&
    navigator.mediaDevices.getUserMedia);
}

function brightnessOff(){

  ctx.filter = 'brightness(0)';
 document.getElementById("webcam").style.filter = 'brightness(0)';

}

function invert(){

  ctx.filter = 'invert(1)';

 document.getElementById("webcam").style.filter = 'invert(1)';

}

function invertandgrayscale(){

  ctx.filter = ' invert(1) grayscale(1)';

 document.getElementById("webcam").style.filter =  ' invert(1) grayscale(1)';

}

function invertandgrayscaleandcontrast(){

  ctx.filter = ' invert(1) grayscale(1) contrast(2)';

 document.getElementById("webcam").style.filter = ' invert(1) grayscale(1) contrast(2)';

}
```

```
function grayscale(){

  ctx.filter = ' grayscale(1)';

 document.getElementById("webcam").style.filter = ' grayscale(1)';

}

function Reset(){

ctx.filter = 'none';

document.getElementById("webcam").style.filter = 'none';

}

// Keep a reference of all the child elements we create
// so we can remove them easilly on each render.
var children = [];

// If webcam supported, add event listener to button for when user
// wants to activate it.

if (hasGetUserMedia()) {

  const enableWebcamButton = document.getElementById('webcamButton3');
  enableWebcamButton.addEventListener('click', enableCam);

} else {
  console.warn('getUserMedia() is not supported by your browser');
}

// Enable the live webcam view and start classification.
function enableCam(event) {
```

```
  if (!model) {
    console.log('Wait! Model not loaded yet.')
    return;
  }


  // Hide the button.
  event.target.classList.add('removed');


  // getUsermedia parameters.
  const constraints = {
    video: true
  };


  // Activate the webcam stream.
  navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
  window.speechSynthesis.cancel();
  video.removeEventListener('loadeddata', intruder);
  video.removeEventListener('loadeddata', aerialobject);



document.getElementById("title1").innerHTML = "All Object Detection";


   video.removeEventListener('loadeddata', aerialobjectd);
  video.removeEventListener('loadeddata', intruderd);
   video.srcObject = stream;
    video.addEventListener('loadeddata', predictWebcam);
    const imageContainers = document.getElementsByClassName('classifyOnClick');
    for (let i = 0; i < imageContainers.length; i++) {
  // Add event listener to the child element whichis the img element.


 imageContainers[i].removeEventListener('click', begin2);
 imageContainers[i].removeEventListener('click', getSmileys3);
   imageContainers[i].removeEventListener('click', getSmileys2);
   imageContainers[i].removeEventListener('click', getSmileys1);
   imageContainers[i].removeEventListener('click', getSmileys);
    imageContainers[i].removeEventListener('click', getSmileys4);


   imageContainers[i].removeEventListener('click', enhanced);




    imageContainers[i].removeEventListener('click', getSmileys);




  imageContainers[i].removeEventListener('click', begin1);
 imageContainers[i].removeEventListener('click', getSmileys3);
   imageContainers[i].removeEventListener('click', getSmileys2);
   imageContainers[i].removeEventListener('click', getSmileys1);
   imageContainers[i].removeEventListener('click', getSmileys);
    imageContainers[i].removeEventListener('click', getSmileys4);


   imageContainers[i].removeEventListener('click', enhanced);




    imageContainers[i].removeEventListener('click', getSmileys);







   imageContainers[i].addEventListener('click', begin);
```

```
 imageContainers[i].addEventListener('click', getSmileys3);
   imageContainers[i].addEventListener('click', getSmileys2);
   imageContainers[i].addEventListener('click', getSmileys1);
   imageContainers[i].addEventListener('click', getSmileys);
    imageContainers[i].addEventListener('click', getSmileys4);

   imageContainers[i].addEventListener('click', enhanced);

    imageContainers[i].addEventListener('click', getSmileys);

function stope1 () {document.getElementById("demo").innerHTML = "X <---Click the X to activate secondary detection";

// Load in dimensions

 }}

  });
}

if (hasGetUserMedia()) {

  const enableWebcamButton4 = document.getElementById('webcamButton4');
  enableWebcamButton4.addEventListener('click', prima);

} else {
  console.warn('getUserMedia() is not supported by your browser');
}

function prima(event) {

// Enable the live webcam view and start classificati
  if (!model) {
    console.log('Wait! Model not loaded yet.')
    return;
  }

  // Hide the button.
  event.target.classList.add('removed');

  // getUsermedia parameters.
  const constraints = {
    video: true
  };

  // Activate the webcam stream.
  navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
 window.speechSynthesis.cancel();
video.removeEventListener('loadeddata', aerialobject);
  video.removeEventListener('loadeddata', intruder);

    video.removeEventListener('loadeddata', predictWebcam);
```

```
    video.srcObject = stream;

    video.addEventListener('loadeddata', predictWebcam1);
    }
);}

function predictWebcam1() {

  document.getElementById("intru1").innerHTML = " ";

  // Now let's start classifying the stream.
  model.detect(video).then(function (predictions) {

    // Remove any highlighting we did previous frame.
    for (let i = 0; i < children.length; i++) {

   liveView.removeChild(children[i]);
    }
    children.splice(0);

    // Now lets loop through predictions and draw them to the live view if
    // they have a high confidence score.

            liveView.appendChild(highlighter);
            liveView.appendChild(p);

        // Store drawn objects in memory so we can delete them next time around.
        children.push(highlighter);
        children.push(p);

    // Call this function again to keep predicting when the browser is ready.

    window.requestAnimationFrame(predictWebcam1);
  });
}

function begin(event){
myInterval = setInterval(function () {
  handleClick(event)}, 6);
  document.getElementById("demo").innerHTML = "X-----Secondary Detection Activating...";
  document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATING";

//when the button is clicked

$('button').click(function () {
  //stop the interval
  clearInterval(myInterval);
```

```
  window.speechSynthesis.pause();
 document.getElementById("demo").innerHTML = "X-------Secondary Detection OFF";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION OFF";
});

}

function begin1(event){
myInterval = setInterval(function () {

  enhancedintruder(event)}, 6);
  document.getElementById("demo").innerHTML = "X-----Secondary Detection Activating...";
  document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATING";

//when the button is clicked



$('button').click(function () {
  //stop the interval
  clearInterval(myInterval);
  window.speechSynthesis.pause();
 document.getElementById("demo").innerHTML = "X-------Secondary Detection OFF";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION OFF";
});

}







function begin2(event){
myInterval = setInterval(function () {
  enhancedaerial(event)}, 6);
document.getElementById("demo").innerHTML = "X-----Secondary Detection Activating...";
document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATING";
//when the button is clicked



$('button').click(function () {
  //stop the interval
  clearInterval(myInterval);
  window.speechSynthesis.pause();
 document.getElementById("demo").innerHTML = "X-------Secondary Detection OFF";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION OFF";
});

}







//set an interval and assign it to the variable: "myInterval"







function handleClick(event) {
```

```
  // We can call model.classify as many times as we like with
  // different image data each time. This returns a promise
  // which we wait to complete and then call a function to
  // print out the results of the prediction.
  model.detect(event.target).then(function (predictions) {
    // Lets write the predictions to a new paragraph element and
    // add it to the DOM.



    for (let n = 0; n < predictions.length; n++) {
      // Description text
        if (predictions[n].score > 0) {
      const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';


      const highlighter = document.createElement('div');
      highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';


           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';


        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';



setInterval(function(){
  event.target.parentNode.removeChild(highlighter);
 event.target.parentNode.removeChild(p);



},9);


     if (event.target.parentNode.appendChild(highlighter)){


    setTimeout(() => {
     setInterval(function(){


  event.target.parentNode.removeChild(highlighter);
  event.target.parentNode.removeChild(p);






},9);
 event.target.parentNode.appendChild(p);
  imageContainers[i].addEventListener('load', handleClick);
},9);
```

```
      }

      if (event.target.parentNode.appendChild(p)){
      setTimeout(() => {
 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);

}, 9); }

 if (event.target.parentNode.removeChild(p)){
      setTimeout(() => {
      document.getElementById("intru1").innerHTML = "";

 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);

}, 3); }

      event.target.parentNode.appendChild(p1);
     event.target.parentNode.appendChild(p);
     event.target.parentNode.appendChild(highlighter);

 document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATED";

      children.push.appendChild(highlighter);
      children.push.appendChild(p);
    }
    }
  window.requestAnimationFrame(handleClick);

  });

}

document.getElementById("demo").innerHTML = "X <---Enable Webcam and Click on the X to activate secondary detection";
 if (hasGetUserMedia()) {

  const enableWebcamButton1 = document.getElementById('webcamButton1');
  enableWebcamButton1.addEventListener('click', intruderd);
} else {
  console.warn('getUserMedia() is not supported by your browser');
}

 function intruderd(event) {
  if (!model) {
    console.log('Wait! Model not loaded yet.')
    return;
  }
```

```
 // Hide the button.
 event.target.classList.add('removed');

 // getUsermedia parameters.
 const constraints = {
   video: true
 };

 // Activate the webcam stream.
 navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
 window.speechSynthesis.cancel();
 document.getElementById("title1").innerHTML = "Intruder Detection";
 video.removeEventListener('loadeddata', predictWebcam);
  video.removeEventListener('loadeddata', aerialobject);
video.removeEventListener('loadeddata', predictWebcam1);
  video.srcObject = stream;
 video.addEventListener('loadeddata', intruder);

   const imageContainers1 = document.getElementsByClassName('classifyOnClick');
   for (let i = 0; i < imageContainers1.length; i++) {
 // Add event listener to the child element whichis the img element.

imageContainers1[i].removeEventListener('click', begin);
 imageContainers1[i].removeEventListener('click', begin2);
imageContainers1[i].removeEventListener('click', getSmileys3);
  imageContainers1[i].removeEventListener('click', getSmileys2);
  imageContainers1[i].removeEventListener('click', getSmileys1);
  imageContainers1[i].removeEventListener('click', getSmileys);
   imageContainers1[i].removeEventListener('click', getSmileys4);

  imageContainers1[i].removeEventListener('click', enhanced);

   imageContainers1[i].removeEventListener('click', getSmileys);

 imageContainers1[i].removeEventListener('click', handleClick);
imageContainers1[i].removeEventListener('click', getSmileys3);
  imageContainers1[i].removeEventListener('click', getSmileys2);
  imageContainers1[i].removeEventListener('click', getSmileys1);
  imageContainers1[i].removeEventListener('click', getSmileys);
   imageContainers1[i].removeEventListener('click', getSmileys4);

  imageContainers1[i].removeEventListener('click', enhanced);

   imageContainers1[i].removeEventListener('click', getSmileys);

 imageContainers1[i].addEventListener('click', begin1);
imageContainers1[i].addEventListener('click', getSmileys3);
  imageContainers1[i].addEventListener('click', getSmileys2);
  imageContainers1[i].addEventListener('click', getSmileys1);
  imageContainers1[i].addEventListener('click', getSmileys);
   imageContainers1[i].addEventListener('click', getSmileys4);

  imageContainers1[i].addEventListener('click', enhanced);
```

```
    imageContainers1[i].addEventListener('click', getSmileys);

function stope1 () {document.getElementById("demo").innerHTML = "X <---Click the X to activate secondary detection";

// Load in dimensions

 }}

  });
}

  function stope () {document.getElementById("demo").innerHTML = "X <---Click the X to activate secondary detection";

  }

  function rests (){
  document.getElementById("randomnumber1").style.visibility="hidden";
  document.getElementById("randomnumber2").style.visibility="hidden";
  document.getElementById("randomnumber3").style.visibility="hidden";
  document.getElementById("randomscan").style.visibility="hidden";
  document.getElementById("randomnumber1").style.visibility="hidden";

document.getElementById("endec").style.visibility="hidden";
 document.getElementById("endec1").style.visibility="hidden";
  document.getElementById("endec2").style.visibility="hidden";

  }

  function rests1() {
  document.getElementById("endec1").style.visibility="hidden";
  document.getElementById("endec2").style.visibility="hidden";
  }

  function rests2() {
  document.getElementById("endec1").style.visibility="true";
  document.getElementById("endec2").style.visibility="true";
  }

  function enhancedintruder (event) {

   model.detect(event.target).then(function (predictions) {
```

```
    // Lets write the predictions to a new paragraph element and
    // add it to the DOM.
window.speechSynthesis.pause();

    for (let n = 0; n < predictions.length; n++) {

if ( predictions[n].class == "person") {

predictions[n].class = "Intruder Detected"
document.getElementById("intru1").innerHTML = "Intruder Detected";
window.speechSynthesis.resume();

textToSpeech();

beep(1000, 2, function () {

});

      // Description text

      const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';

      const highlighter = document.createElement('div');
      highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';

           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';

setInterval(function(){
  event.target.parentNode.removeChild(highlighter);
```

```
 event.target.parentNode.removeChild(p);



},9);

     if (event.target.parentNode.appendChild(highlighter)){

    setTimeout(() => {
     setInterval(function(){

  event.target.parentNode.removeChild(highlighter);
  event.target.parentNode.removeChild(p);




},9);
 event.target.parentNode.appendChild(p);
  imageContainers[i].addEventListener('load', handleClick);
},9);





      }





      if (event.target.parentNode.appendChild(p)){
      setTimeout(() => {
 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 9); }

 if (event.target.parentNode.removeChild(p)){
      setTimeout(() => {
      document.getElementById("intru1").innerHTML = "";


 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 3); }








      event.target.parentNode.appendChild(p1);
     event.target.parentNode.appendChild(p);
     event.target.parentNode.appendChild(highlighter);









 document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATED";




      children.push.appendChild(highlighter);
      children.push.appendChild(p);
    }
    }
```

```
  window.requestAnimationFrame(enhancedintruder);

  });
}

  function intruder() {

   document.getElementById("intru2").innerHTML = " ";

model.detect(video).then(function (predictions) {

for (let i = 0; i < children.length; i++) {

liveView.removeChild(children[i]);
document.getElementById("intru1").innerHTML = " ";

}

children.splice(0);
window.speechSynthesis.pause();

for (let n = 0; n < predictions.length; n++) {

if ( predictions[n].class == "person") {

predictions[n].class = "Intruder Detected"

document.getElementById("intru1").innerHTML = "Intruder Detected";

window.speechSynthesis.resume();

textToSpeech();

beep(1000, 2, function () {

});

const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
```

```
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';


        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';


liveView.appendChild(highlighter);


liveView.appendChild(p);





children.push(highlighter);


children.push(p);




}


else{


}


}


window.requestAnimationFrame(intruder);


});






}






if (hasGetUserMedia()) {


  const enableWebcamButton2 = document.getElementById('webcamButton2');
  enableWebcamButton2.addEventListener('click', aerialobjectd);
} else {
  console.warn('getUserMedia() is not supported by your browser');
}


function aerialobjectd(event) {
  if (!model) {
    console.log('Wait! Model not loaded yet.')
    return;
  }


  // Hide the button.
  event.target.classList.add('removed');


  // getUsermedia parameters.
  const constraints = {
    video: true
  };


  // Activate the webcam stream.
  navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
```

```
  window.speechSynthesis.cancel();
 video.removeEventListener('loadeddata', intruder);
 document.getElementById("title1").innerHTML = "Aerial Object Detection";
 video.removeEventListener('loadeddata', predictWebcam);
  video.removeEventListener('loadeddata', aerialobjectd);
  video.removeEventListener('loadeddata', predictWebcam1);
 video.removeEventListener('loadeddata', intruderd);
  video.srcObject = stream;
  video.addEventListener('loadeddata', aerialobject);
   const imageContainers2 = document.getElementsByClassName('classifyOnClick');
   for (let i = 0; i < imageContainers2.length; i++) {
 // Add event listener to the child element whichis the img element.
 imageContainers2[i].removeEventListener('click', begin);
imageContainers2[i].removeEventListener('click', getSmileys3);
  imageContainers2[i].removeEventListener('click', getSmileys2);
  imageContainers2[i].removeEventListener('click', getSmileys1);
  imageContainers2[i].removeEventListener('click', getSmileys);
   imageContainers2[i].removeEventListener('click', getSmileys4);

  imageContainers2[i].removeEventListener('click', enhanced);

   imageContainers2[i].removeEventListener('click', getSmileys);

 imageContainers2[i].removeEventListener('click', begin1);
imageContainers2[i].removeEventListener('click', getSmileys3);
  imageContainers2[i].removeEventListener('click', getSmileys2);
  imageContainers2[i].removeEventListener('click', getSmileys1);
  imageContainers2[i].removeEventListener('click', getSmileys);
   imageContainers2[i].removeEventListener('click', getSmileys4);

  imageContainers2[i].removeEventListener('click', enhanced);

   imageContainers2[i].removeEventListener('click', getSmileys);

 imageContainers2[i].addEventListener('click', begin2);
imageContainers2[i].addEventListener('click', getSmileys3);
  imageContainers2[i].addEventListener('click', getSmileys2);
  imageContainers2[i].addEventListener('click', getSmileys1);
  imageContainers2[i].addEventListener('click', getSmileys);
   imageContainers2[i].addEventListener('click', getSmileys4);

  imageContainers2[i].addEventListener('click', enhanced);

   imageContainers2[i].addEventListener('click', getSmileys);

function stope1 () {document.getElementById("demo").innerHTML = "X <---Click the X to activate secondary detection";

// Load in dimensions
```

```
 }}

  });
}

  function enhancedaerial (event){
  document.getElementById("intru1").innerHTML = "";

  model.detect(event.target).then(function (predictions) {
    // Lets write the predictions to a new paragraph element and
    // add it to the DOM.

   window.speechSynthesis.pause();
    for (let n = 0; n < predictions.length; n++) {
     if ( predictions[n].class == "knife") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();

textToSpeech();

beep(1000, 2, function () {

});

  const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';

      const highlighter = document.createElement('div');
      highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';

           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';

setInterval(function(){
  event.target.parentNode.removeChild(highlighter);
 event.target.parentNode.removeChild(p);

},9);
```

```
     if (event.target.parentNode.appendChild(highlighter)){

    setTimeout(() => {
     setInterval(function(){

  event.target.parentNode.removeChild(highlighter);
  event.target.parentNode.removeChild(p);

},9);
 event.target.parentNode.appendChild(p);
  imageContainers[i].addEventListener('load', handleClick);
},9);

      }

      if (event.target.parentNode.appendChild(p)){
      setTimeout(() => {
 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);

}, 9); }

 if (event.target.parentNode.removeChild(p)){
      setTimeout(() => {
      document.getElementById("intru1").innerHTML = "";

 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);

}, 3); }

      event.target.parentNode.appendChild(p1);
     event.target.parentNode.appendChild(p);
     event.target.parentNode.appendChild(highlighter);

 document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";

      children.push.appendChild(highlighter);
      children.push.appendChild(p);

    }
     if ( predictions[n].class == "remote") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();
```

```
textToSpeech();

beep(1000, 2, function () {

});

 const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';

      const highlighter = document.createElement('div');
      highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';

           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';

setInterval(function(){
  event.target.parentNode.removeChild(highlighter);
 event.target.parentNode.removeChild(p);

},9);

     if (event.target.parentNode.appendChild(highlighter)){

    setTimeout(() => {
     setInterval(function(){

  event.target.parentNode.removeChild(highlighter);
  event.target.parentNode.removeChild(p);

},9);
 event.target.parentNode.appendChild(p);
  imageContainers[i].addEventListener('load', handleClick);
},9);

      }
```

```
      if (event.target.parentNode.appendChild(p)){
      setTimeout(() => {
 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);

}, 9); }

 if (event.target.parentNode.removeChild(p)){
      setTimeout(() => {
      document.getElementById("intru1").innerHTML = "";

 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);

}, 3); }

      event.target.parentNode.appendChild(p1);
     event.target.parentNode.appendChild(p);
     event.target.parentNode.appendChild(highlighter);

 document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATED";

      children.push.appendChild(highlighter);
      children.push.appendChild(p);

    }
     if ( predictions[n].class == "frisbee") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();

textToSpeech();

beep(1000, 2, function () {

});

 const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';

      const highlighter = document.createElement('div');
      highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
```

```
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';


           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';


        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';


setInterval(function(){
  event.target.parentNode.removeChild(highlighter);
 event.target.parentNode.removeChild(p);




},9);


     if (event.target.parentNode.appendChild(highlighter)){


    setTimeout(() => {
     setInterval(function(){


  event.target.parentNode.removeChild(highlighter);
  event.target.parentNode.removeChild(p);






},9);
 event.target.parentNode.appendChild(p);
  imageContainers[i].addEventListener('load', handleClick);
},9);






      }







      if (event.target.parentNode.appendChild(p)){
      setTimeout(() => {
 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 9); }


 if (event.target.parentNode.removeChild(p)){
      setTimeout(() => {
      document.getElementById("intru1").innerHTML = "";


 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 3); }
```

```
      event.target.parentNode.appendChild(p1);
     event.target.parentNode.appendChild(p);
     event.target.parentNode.appendChild(highlighter);

 document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATED";

      children.push.appendChild(highlighter);
      children.push.appendChild(p);
    }

     if ( predictions[n].class == "airplane") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();

textToSpeech();

beep(1000, 2, function () {

});

 const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';

      const highlighter = document.createElement('div');
      highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';

           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';

setInterval(function(){
```

```
 event.target.parentNode.removeChild(highlighter);
event.target.parentNode.removeChild(p);



},9);

    if (event.target.parentNode.appendChild(highlighter)){

   setTimeout(() => {
    setInterval(function(){

 event.target.parentNode.removeChild(highlighter);
 event.target.parentNode.removeChild(p);




},9);
event.target.parentNode.appendChild(p);
 imageContainers[i].addEventListener('load', handleClick);
},9);




     }





     if (event.target.parentNode.appendChild(p)){
     setTimeout(() => {
event.target.parentNode.removeChild(p);
 imageContainers[i].addEventListener('load', handleClick);

}, 9); }

if (event.target.parentNode.removeChild(p)){
     setTimeout(() => {
     document.getElementById("intru1").innerHTML = "";

event.target.parentNode.removeChild(p);
 imageContainers[i].addEventListener('load', handleClick);

}, 3); }







     event.target.parentNode.appendChild(p1);
    event.target.parentNode.appendChild(p);
    event.target.parentNode.appendChild(highlighter);







document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";
document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATED";



     children.push.appendChild(highlighter);
     children.push.appendChild(p);
   }
```

```
    if ( predictions[n].class == "kite") {


predictions[n].class = "Aerial Object Detected"
document.getElementById("intru").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();


textToSpeech();


beep(1000, 2, function () {


});
 const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';


      const highlighter = document.createElement('div');
      highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';


           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';


        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';


setInterval(function(){
  event.target.parentNode.removeChild(highlighter);
 event.target.parentNode.removeChild(p);



},9);


     if (event.target.parentNode.appendChild(highlighter)){


    setTimeout(() => {
     setInterval(function(){


  event.target.parentNode.removeChild(highlighter);
  event.target.parentNode.removeChild(p);





},9);
 event.target.parentNode.appendChild(p);
  imageContainers[i].addEventListener('load', handleClick);
},9);
```

```
      }




      if (event.target.parentNode.appendChild(p)){
      setTimeout(() => {
 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 9); }


 if (event.target.parentNode.removeChild(p)){
      setTimeout(() => {
      document.getElementById("intru1").innerHTML = "";


 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 3); }





      event.target.parentNode.appendChild(p1);
     event.target.parentNode.appendChild(p);
     event.target.parentNode.appendChild(highlighter);






 document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATED";


      children.push.appendChild(highlighter);
      children.push.appendChild(p);
    }


 if ( predictions[n].class == "bird") {


predictions[n].class = "Aerial Object Detected"
document.getElementById("intru").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();


textToSpeech();


beep(1000, 2, function () {


});


 const p = document.createElement('p');
      p.innerText = predictions[n].class  + ' - with '
          + Math.round(parseFloat(predictions[n].score) * 100)
          + '% confidence.';
      // Positioned at the top left of the bounding box.
      // Height is whatever the text takes up.
      // Width subtracts text padding in CSS so fits perfectly.
      p.style = 'left: ' + predictions[n].bbox[2] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[0] - 10) + 'px;';


      const highlighter = document.createElement('div');
```

```
     highlighter.setAttribute('class', 'highlighter');
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px;' +
          'width: ' + predictions[n].bbox[2] + 'px;' +
          'height: ' + predictions[n].bbox[3] + 'px;';
         p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
          'top: ' + predictions[n].bbox[1] + 'px; ' +
          'width: ' + (predictions[n].bbox[2] - 5) + 'px;';


           const p1 = document.createElement('h1');
        p1.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p1.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';


        // Draw the actual bounding box.
        const highlighter1 = document.createElement('div');
        highlighter1.setAttribute('class', 'highlighter1');
        highlighter1.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';


setInterval(function(){
  event.target.parentNode.removeChild(highlighter);
 event.target.parentNode.removeChild(p);




},9);


     if (event.target.parentNode.appendChild(highlighter)){


    setTimeout(() => {
     setInterval(function(){


  event.target.parentNode.removeChild(highlighter);
  event.target.parentNode.removeChild(p);






},9);
 event.target.parentNode.appendChild(p);
  imageContainers[i].addEventListener('load', handleClick);
},9);






      }







      if (event.target.parentNode.appendChild(p)){
      setTimeout(() => {
 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 9); }


 if (event.target.parentNode.removeChild(p)){
      setTimeout(() => {
      document.getElementById("intru1").innerHTML = "";


 event.target.parentNode.removeChild(p);
  imageContainers[i].addEventListener('load', handleClick);


}, 3); }
```

```
      event.target.parentNode.appendChild(p1);
     event.target.parentNode.appendChild(p);
     event.target.parentNode.appendChild(highlighter);

 document.getElementById("demo").innerHTML = "X  Secondary Detection Activated";
 document.getElementById("endec").innerHTML = "SECONDARY DETECTION ACTIVATED";

      children.push.appendChild(highlighter);
      children.push.appendChild(p);
    }
    }

  window.requestAnimationFrame(enhancedaerial);

  });

}

function aerialobject() {
document.getElementById("intru1").innerHTML = " ";

  document.getElementById("intru2").innerHTML = " ";

model.detect(video).then(function (predictions) {

for (let i = 0; i < children.length; i++) {

liveView.removeChild(children[i]);
document.getElementById("intru2").innerHTML = " ";
}

children.splice(0);

window.speechSynthesis.pause();
```

```
for (let n = 0; n < predictions.length; n++) {
      // If we are over 66% sure we are sure we classified it right, draw it!

if ( predictions[n].class == "bird") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru2").innerHTML = "Aerial Object Detected";

window.speechSynthesis.resume();
textToSpeech1();

beep(1000, 2, function () {

});

        const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';

liveView.appendChild(highlighter);

liveView.appendChild(p);

children.push(highlighter);

children.push(p);

}

else{

}

if ( predictions[n].class == "kite") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru2").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();
```

```
textToSpeech1();

beep(1000, 2, function () {

});





const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';


liveView.appendChild(highlighter);

liveView.appendChild(p);

children.push(highlighter);

children.push(p);


}

else{

}





if ( predictions[n].class == "frisbee") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru2").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();

textToSpeech1();

beep(1000, 2, function () {

});
```

```
const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';

liveView.appendChild(highlighter);

liveView.appendChild(p);

children.push(highlighter);

children.push(p);

}

else{

}

if ( predictions[n].class == "remote") {

predictions[n].class = "Aerial Object Detected"
document.getElementById("intru2").innerHTML = "Aerial Object Detected";

window.speechSynthesis.resume();

textToSpeech1();

beep(1000, 2, function () {

});

const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
```

```
            + predictions[n].bbox[3] + 'px;';



liveView.appendChild(highlighter);

liveView.appendChild(p);

children.push(highlighter);

children.push(p);


}

else{

}



if ( predictions[n].class == "knife") {

predictions[n].class = "Aerial Object Detected"

document.getElementById("intru2").innerHTML = "Aerial Object Detected";

window.speechSynthesis.resume();

textToSpeech1();

beep(1000, 2, function () {

});


const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';


        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';

liveView.appendChild(highlighter);

liveView.appendChild(p);

children.push(highlighter);

children.push(p);


}
```

```
else{

}


if ( predictions[n].class == "airplane") {

predictions[n].class = "Aerial Object Detected"


document.getElementById("intru2").innerHTML = "Aerial Object Detected";
window.speechSynthesis.resume();

textToSpeech1();

beep(1000, 2, function () {

});


const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';



liveView.appendChild(highlighter);

liveView.appendChild(p);

children.push(highlighter);

children.push(p);


}

else{

}

}

window.requestAnimationFrame(aerialobject);

});

}
```

```
function predictWebcam() {

  document.getElementById("intru1").innerHTML = " ";



  // Now let's start classifying the stream.
  model.detect(video).then(function (predictions) {




    // Remove any highlighting we did previous frame.
    for (let i = 0; i < children.length; i++) {

   liveView.removeChild(children[i]);
    }
    children.splice(0);

    // Now lets loop through predictions and draw them to the live view if
    // they have a high confidence score.
    for (let n = 0; n < predictions.length; n++) {
      // If we are over 66% sure we are sure we classified it right, draw it!
      if (predictions[n].score > 0.66) {



        const p = document.createElement('p');
        p.innerText = predictions[n].class  + ' - with '
            + Math.round(parseFloat(predictions[n].score) * 100)
            + '% confidence.';
        // Draw in top left of bounding box outline.
        p.style = 'left: ' + predictions[n].bbox[0] + 'px;' +
            'top: ' + predictions[n].bbox[1] + 'px;' +
            'width: ' + (predictions[n].bbox[2] - 10) + 'px;';

        // Draw the actual bounding box.
        const highlighter = document.createElement('div');
       highlighter.setAttribute('class', 'highlighter');
        p.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
            + predictions[n].bbox[1] + 'px; width: '
            + predictions[n].bbox[2] + 'px; height: '
            + predictions[n].bbox[3] + 'px;';






            liveView.appendChild(highlighter);
            liveView.appendChild(p);





        // Store drawn objects in memory so we can delete them next time around.
        children.push(highlighter);
        children.push(p);


      }
    }

    // Call this function again to keep predicting when the browser is ready.

    window.requestAnimationFrame(predictWebcam);
  });
}



function getSmileys() {
  // It's a div, not a button
  var div = document.getElementById("randomnumber1");
  // ** Get the *computed* style of the div
  var style = getComputedStyle(div);
```

```
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
  }
  else {
    div.style.visibility = 'visible'
  }
}


function getSmileysa1() {
  // It's a div, not a button
  var div = document.getElementById("randomnumber1");
  // ** Get the *computed* style of the div
  var style = getComputedStyle(div);
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
  }
  else {
    div.style.visibility = 'visible'
  }
}


function getSmileys4() {
  // It's a div, not a button
  var div = document.getElementById("endec");
  // ** Get the *computed* style of the div
  var style = getComputedStyle(div);
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
  }
  else {
    div.style.visibility = 'visible'
  }
}


function getSmileys5() {
  // It's a div, not a button
  var div = document.getElementById("endec1");
  // ** Get the *computed* style of the div
 var style = getComputedStyle(div);
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
  }
  else {
    div.style.visibility = 'visible'
  }
}


function getSmileys6() {
  // It's a div, not a button
  var div = document.getElementById("endec2");
  // ** Get the *computed* style of the div
  var style = getComputedStyle(div);
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
  }
  else {
    div.style.visibility = 'visible'
  }
}



function getSmileys1() {
  // It's a div, not a button
  var div = document.getElementById("randomnumber2");
  // ** Get the *computed* style of the div
  var style = getComputedStyle(div);
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
  }
  else {
    div.style.visibility = 'visible'
  }
}
function getSmileys2() {
  // It's a div, not a button
  var div = document.getElementById("randomnumber3");
  // ** Get the *computed* style of the div
  var style = getComputedStyle(div);
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
```

```
  }
  else {
    div.style.visibility = 'visible'
  }
}


function getSmileys3() {
  // It's a div, not a button
  var div = document.getElementById("randomscan");
  // ** Get the *computed* style of the div
  var style = getComputedStyle(div);
  if (style.visibility == 'hidden') {
    div.style.visibility = 'true'
  }
  else {
    div.style.visibility = 'visible'
  }
}







 function getDateTime() {
        var now     = new Date();
        var year    = now.getFullYear();
        var month   = now.getMonth()+1;
        var day     = now.getDate();
        var hour    = now.getHours();
        var minute  = now.getMinutes();
        var second  = now.getSeconds();
        if(month.toString().length == 1) {
             month = '0'+month;
        }
        if(day.toString().length == 1) {
             day = '0'+day;
        }
        if(hour.toString().length == 1) {
             hour = '0'+hour;
        }
        if(minute.toString().length == 1) {
             minute = '0'+minute;
        }
        if(second.toString().length == 1) {
             second = '0'+second;
        }
        var dateTime = year+'-'+month+'-'+day+' '+hour+':'+minute+':'+second;
         return dateTime;
    }


    // example usage: realtime clock
    setInterval(function(){
        currentTime = getDateTime();
        document.getElementById("digital-clock").innerHTML = currentTime;
    }, 1000);


   let batteryPromise = navigator.getBattery();
batteryPromise.then(batteryCallback);


function batteryCallback(batteryObject) {
   printBatteryStatus(batteryObject);
}
function printBatteryStatus(batteryObject) {
    console.log("IsCharging", batteryObject.charging);
    console.log("Percentage", batteryObject.level);


    console.log("charging Time", batteryObject.chargingTime);
    console.log("DisCharging Time", batteryObject.dischargingTime);
}


    </script>
<script>
function enhanced(){
  var lFollowX = 0;
    var lFollowY = 0;
    var x = 0;
    var y = 0;
```

```
    var friction = 1 / 30;

function moveBackground() {

x += (lFollowX - x) * friction;
y += (lFollowY - y) * friction;

$('.positionx').text(x);
$('.positiony').text(y);

translate = 'translate(' + x + 'px, ' + y + 'px) scale(1.2)';

$('.stage').css({
'-webit-transform': translate,
'-moz-transform': translate,
'transform': translate
});

window.requestAnimationFrame(moveBackground);

}

$(window).on('mousemove click', function(e) {

var lMouseX = Math.max(-100, Math.min(100, $(window).width() / 2 - e.clientX));
var lMouseY = Math.max(-100, Math.min(100, $(window).height() / 2 - e.clientY));
lFollowX = (200 * lMouseX) / 100;
lFollowY = (80 * lMouseY) / 100;

});

moveBackground();



// Random number generator
setInterval(function(){
ChangeNumber1();
ChangeNumber2();
ChangeNumber3();
}, 1000);
function ChangeNumber1() {

var newNumber = Math.floor(Math.random(9) * 1000000);
$('#randomnumber1').text(newNumber);
}
function ChangeNumber2() {
var newNumber = Math.floor(Math.random(9) * 100000000);
$('#randomnumber2').text(newNumber);
}
function ChangeNumber3() {
var newNumber = Math.floor(Math.random(9) * 10000000000);
$('#randomnumber3').text(newNumber);
}

setInterval(function(){
ChangeNumber4();
}, 1500);
function ChangeNumber4() {
var newNumber = Math.floor(Math.random(9) * 100000);
$('#randomscan').text(newNumber);
}

// Load in dimensions
setTimeout(function(){ $('.dimension1').addClass('show') }, 1000);
setTimeout(function(){ $('.dimension2').addClass('show') }, 2000);
setTimeout(function(){ $('.dimension3').addClass('show') }, 3000);
setTimeout(function(){ $('.dimension4').addClass('show') }, 4000);
setTimeout(function(){ $('.dimension5').addClass('show') }, 5000);
}
```

```
   </script>

  </body>
</html>
```